大数据管理方法与应用第二次作业

大数据 001

郅啸淇

学号：2184114639

2023 年 4 月 2 日

1 鸢尾花数据集降维

1.1 代码思路

首先导入 Iris 数据集，然后使用 PCA 方法进行降维，保留三个主成分，最后使用 Axes3D 进行绘图，如图 1 所示。

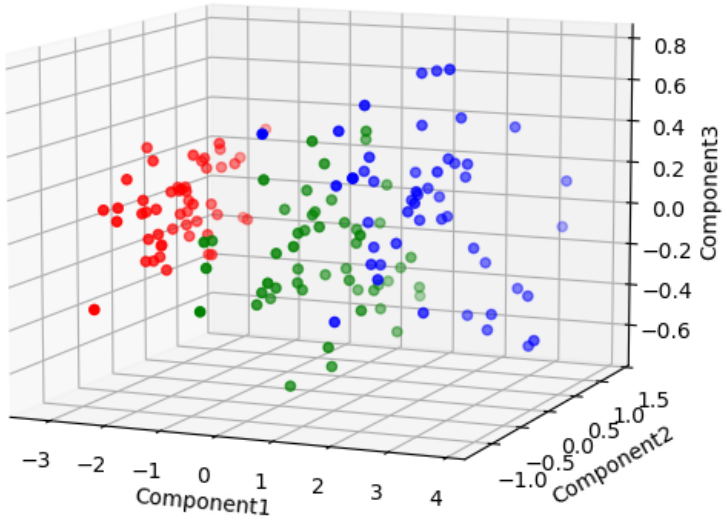

图 1: 降维后鸢尾花数据集

1.2 完整代码

```
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
```

```
7
8  iris = load_iris()
9  # 确定函数和函数方法
10 pca = PCA(n_components=3)
11 data = pca.fit_transform(iris.data)
12
13 # 取出各类样本对应样本
14 setosa = np.where(iris.target == 0)
15 versicolor = np.where(iris.target == 1)
16 virginica = np.where(iris.target == 2)
17 labels = ["setosa", "versicolor", "virginica"]
18
19
20 fig = plt.figure()
21 ax = Axes3D(fig)
22
23 ax.scatter(data[setosa][:, 0], data[setosa][:, 1], data[setosa][:, 2],c = 'r')
24 ax.scatter(data[versicolor][:, 0], data[versicolor][:, 1], data[versicolor][:, 2],c =
       'g')
25 ax.scatter(data[virginica][:, 0], data[virginica][:, 1], data[virginica][:, 2],c = 'b'
       )
26 ax.set_xlabel('Component1')
27 ax.set_ylabel('Component2')
28 ax.set_zlabel('Component3')
29 plt.show()
```

2 手写体数据集降维分类

2.1 代码思路

首先不使用 PCA 对数据降维，直接使用 SVM 进行分类，得出正确率为 0.9867，使用 PCA 压缩保留前 32 个主成分（原图片为 8*8，共 64 个维度），得到新的分类正确率为 0.5089

2.2 完整代码

```
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn import svm
import matplotlib.pyplot as plt
digits = load_digits()
# 原矩阵为8*8
# PCA压缩前分类
Xtrain, Xtest, Ytrain, Ytest = train_test_split(digits.data, digits.target, test_size
    =0.25, random_state=114514)
clf = svm.SVC(gamma=0.001, C=100)
clf.fit(Xtrain, Ytrain)
score1 = clf.score(Xtest,Ytest)

fig, axes = plt.subplots(4, 4, figsize=(8,8))
fig.subplots_adjust(hspace=0.1, wspace=0.1)

Ypred = clf.predict(Xtest)

for i, ax in  enumerate(axes.flat):
    ax.imshow(Xtest[i].reshape(8,8), cmap=plt.cm.gray_r, interpolation='nearest')
    ax.text(0.05, 0.05,  str(Ypred[i]), fontsize=32, transform=ax.transAxes, color='
        green' if Ypred[i] == Ytest[i] else 'red')
    ax.text(0.8, 0.05,
        str(Ytest[i]), fontsize=32, transform=ax.transAxes, color='black')
    ax.set_xticks([])
    ax.set_yticks([])
plt.show()
print("PCA压缩前测试集分类正确率:",score1)

# PCA压缩后分类

pca = PCA(n_components=32)
X_trans_train = pca.fit_transform(Xtrain)
X_trans_test = pca.fit_transform(Xtest)
```

```
clf = svm.SVC(gamma=0.001, C=100)
clf.fit(X_trans_train, Ytrain)
score2 = clf.score(X_trans_test,Ytest)

fig, axes = plt.subplots(4, 4, figsize=(8,8))
fig.subplots_adjust(hspace=0.1, wspace=0.1)

Ypred = clf.predict(X_trans_test)

for i, ax in  enumerate(axes.flat):
    ax.imshow(Xtest[i].reshape(8,8), cmap=plt.cm.gray_r, interpolation='nearest')
    ax.text(0.05, 0.05,  str(Ypred[i]), fontsize=32, transform=ax.transAxes, color='
        green' if Ypred[i] == Ytest[i] else 'red')
    ax.text(0.8, 0.05,
        str(Ytest[i]), fontsize=32, transform=ax.transAxes, color='black')
    ax.set_xticks([])
    ax.set_yticks([])
plt.show()

print("PCA压缩后测试集分类正确率:",score2)
```

3 PCA 压缩图片

3.1 代码思路

首先使用 PIL 读入图片，原图片如图 2 所示，其分辨率为 555*377，接着将每一行视作一个特征，使用 PCA 分别保留 1%，2%，5%，10%，20%，30%，如图 3，4，5，6，7，8 所示，可以看出保留比例越大图像效果越接近原图。

图 2: 原图

图 3: 保留 1%

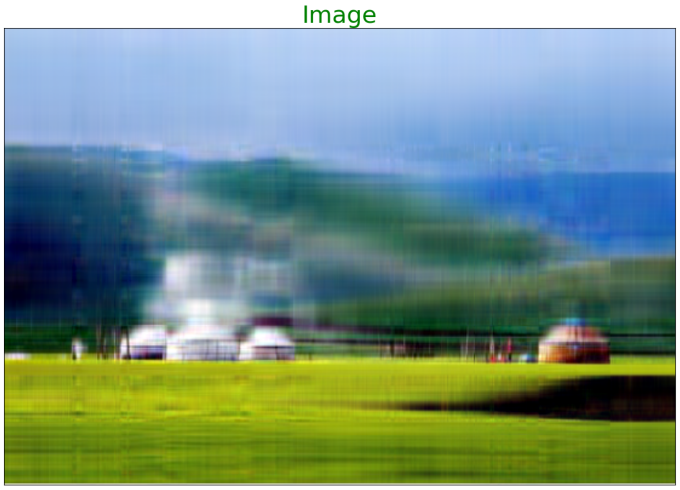

图 4: 保留 2%

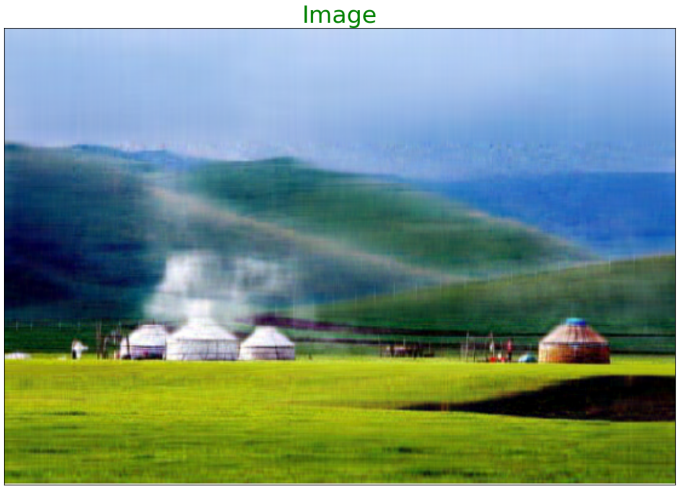

图 5: 保留 5%

图 6: 保留 10%

图 7: 保留 20%

图 8: 保留 30%

3.2 完整代码

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io
from sklearn.decomposition import PCA

A = io.imread('test.jpg')
A = A / 255  # RGB的三个值[0,255]，将它们范围设置为[0,1]

R, G, B = A[:, :, 0], A[:, :, 1], A[:, :, 2]


def pca(X, K):
    pca = PCA(n_components=K).fit(X)
    X_new = pca.transform(X)
    X_new = pca.inverse_transform(X_new)
    return X_new


```

```
ratio = [0.01, 0.02, 0.05, 0.1, 0.2, 0.3]
index = []
for i in ratio:
    index.append( int(555 * i))

for k in index:
    R_new, G_new, B_new = pca(R, k), pca(G, k), pca(B, k)

    A_new = np.zeros(A.shape)
    A_new[:, :, 0] = R_new
    A_new[:, :, 1] = G_new
    A_new[:, :, 2] = B_new

    fig, ax_array = plt.subplots(nrows=1, ncols=1, figsize=(64, 64))
    cmap_list = ['Reds', 'Greens', 'Blues']
    ax_array.imshow(A_new[:, :, :])
    ax_array.set_xticks([])
    ax_array.set_yticks([])
    ax_array.set_title("Image", size=30, color='g')
    plt.show()
```

4 人脸识别数据集降维分类

4.1 代码思路

首先读取 fetch_lfw_people 数据集，直接使用 SVM 进行分类，正确率为 0.80，然后使用 PCA 方法保留前 150 个主成分，再进行 SVM 分类，正确率降低为 0.77，同时提取出前十二个权重最大的主成分并绘图，得到权重排名前十二的特征脸，如图 9 所示。

图 9: 权重排名前十二的特征脸

4.2 完整代码

```
from sklearn.datasets import fetch_lfw_people
from sklearn import svm
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

pca = PCA(n_components=150,whiten=True,random_state=114514)
svc = svm.SVC(kernel='linear', class_weight='balanced')
model1 = make_pipeline(pca, svc)
model2 = make_pipeline(svc)

faces = fetch_lfw_people(min_faces_per_person=60)
faces_data = faces.data
Xtrain, Xtest, Ytrain, Ytest = train_test_split(faces.data, faces.target, random_state
```

```
    =114514)
19
20  param_grid = {'svc__C': [1, 5, 10, 50]}
21  grid1 = GridSearchCV(model1,param_grid)
22  grid2 = GridSearchCV(model2,param_grid)
23
24  grid1.fit(Xtrain,Ytrain)
25  grid2.fit(Xtrain,Ytrain)
26
27  print(grid1.best_params_)
28  print(grid2.best_params_)
29
30  model1 = grid1.best_estimator_
31  model2 = grid2.best_estimator_
32  yfit1 = model1.predict(Xtest)
33  yfit2 = model2.predict(Xtest)
34  print("---------------PCA后SVM---------------")
35  print(classification_report(Ytest,yfit1,target_names=faces.target_names))
36  print("--------------直接SVM--------------")
37  print(classification_report(Ytest,yfit2,target_names=faces.target_names))
38
39  def plot_digits(data):
40      fig,axes = plt.subplots(4,3,figsize=(15,15),
41      subplot_kw={'xticks':[],'yticks':[]},
42      gridspec_kw= dict(hspace=0.1,wspace=0.1))
43      for i,ax in  enumerate(axes.flat):
44          ax.imshow(data[i].reshape(62,47),cmap='bone')
45      plt.show()
46
47  pca2 = PCA(n_components=150)
48  pca2.fit(faces_data)
49  plot_digits(pca2.components_)
```